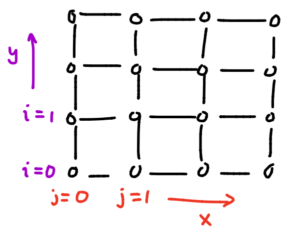
3.5) IMAGE COMPRESSION

The DCT lies at the heart of modern compression techniques for images and audio (e.g. JPEG, MPEG).

For images, we have a 2-d grid of pixel values (greyscale or colour intensities), so we will need the 2-d DCT. This is simply the 1-d DCT applied in two dimensions, one after the other.



Let $X = (x_{ij})$ be the matrix of pixel values.

- First apply the DCT in the x-direction:

$$C_n^{-1} X^T \longrightarrow \text{resulting columns are DCTs of the rows of } X \text{ (fixed } y_i\text{).}$$

- Now apply a DCT in the y-direction:

$$C_n^{-1}(C_n^{-1}X^T)^T = C_n^{-1}X(C_n^{-1})^T = C_n^{-1}XC_n.$$

i.e. the 2-d DCT of a matrix X is $Y = C_n^{-1}XC_n$.

Example:- Find the 2-d DCT of the data

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

The $n=4$ matrix is

$$C_4 = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos(\frac{\pi}{8}) & \cos(\frac{2\pi}{8}) & \cos(\frac{3\pi}{8}) \\ \frac{1}{\sqrt{2}} & \cos(\frac{3\pi}{8}) & \cos(\frac{6\pi}{8}) & \cos(\frac{9\pi}{8}) \\ \frac{1}{\sqrt{2}} & \cos(\frac{5\pi}{8}) & \cos(\frac{10\pi}{8}) & \cos(\frac{15\pi}{8}) \\ \frac{1}{\sqrt{2}} & \cos(\frac{7\pi}{8}) & \cos(\frac{14\pi}{8}) & \cos(\frac{21\pi}{8}) \end{bmatrix} = a \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

where

$$a = \frac{1}{\sqrt{2}}, \quad b = \cos(\tfrac{\pi}{8}), \quad c = \cos(\tfrac{3\pi}{8}).$$

The 2-d DCT is

$$Y = C_4^{-1}XC_4 = a^2 \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & b & a & -c \end{bmatrix} = \begin{bmatrix} 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

It follows that the inverse transform is $X = C_n Y C_n^{-1}$.

Note that

$$(C_n)_{ij} = \sqrt{\tfrac{2}{n}}\,\sigma_j \cos\left(\tfrac{j\pi}{n}(i+\tfrac{1}{2})\right) \qquad \text{where} \quad \sigma_j = \begin{cases} 1/\sqrt{2} & \text{if } j=0 \\ 1 & \text{if } j>0. \end{cases}$$

just to make notation nicer.

Hence

$$\begin{aligned} x_{ij} &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} (C_n)_{ik}\, y_{kl}\, (C_n^{-1})_{lj} \\ &= \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} (C_n)_{ik}\, y_{kl}\, (C_n)_{jl} \\ &= \frac{2}{n}\sum_{k=0}^{n-1}\sum_{l=0}^{n-1} y_{kl}\,\sigma_k\sigma_l \cos\left(\tfrac{k\pi}{n}(i+\tfrac{1}{2})\right)\cos\left(\tfrac{l\pi}{n}(j+\tfrac{1}{2})\right). \end{aligned}$$

In other words, the y_{kl} are interpolation coefficients of a 2-d trigonometric polynomial

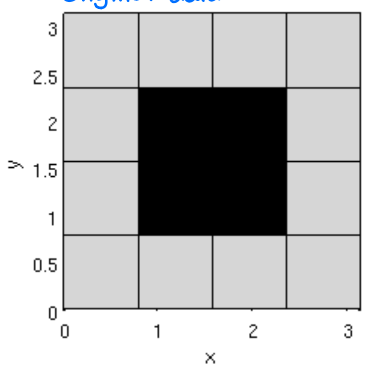
$$p_n(x,y) = \frac{2}{n}\sum_{k=0}^{n-1}\sum_{l=0}^{n-1} y_{kl}\,\sigma_k\sigma_l \cos(lx)\cos(ky)$$

for nodes (x_j, y_i) given by $x_j = \frac{\pi}{n}(j+\frac{1}{2})$, $y_i = \frac{\pi}{n}(i+\frac{1}{2})$.

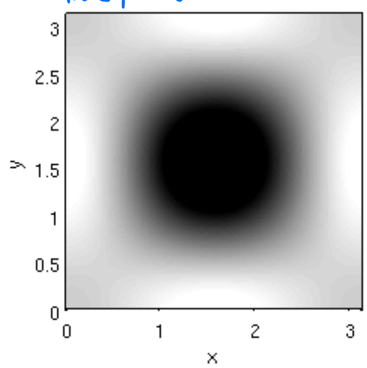
Example :- The interpolation function for the previous example is

$$\begin{aligned} p_4(x,y) = \frac{1}{2}\Big\{ & \frac{1}{2}y_{00} + \frac{1}{\sqrt{2}}y_{01}\cos(x) + \frac{1}{\sqrt{2}}y_{02}\cos(2x) + \frac{1}{\sqrt{2}}y_{03}\cos(3x) \\ & + \frac{1}{\sqrt{2}}y_{10}\cos(y) + y_{11}\cos(x)\cos(y) + y_{12}\cos(2x)\cos(y) + y_{13}\cos(3x)\cos(y) \\ & + \frac{1}{\sqrt{2}}y_{20}\cos(2y) + y_{21}\cos(x)\cos(2y) + y_{22}\cos(2x)\cos(2y) + y_{23}\cos(3x)\cos(2y) \\ & + \frac{1}{\sqrt{2}}y_{30}\cos(3y) + y_{31}\cos(x)\cos(3y) + y_{32}\cos(2x)\cos(3y) + y_{33}\cos(3x)\cos(3y)\Big\}. \\ = \frac{3}{4} & + \frac{1}{2\sqrt{2}}\cos(2x) + \frac{1}{2\sqrt{2}}\cos(2y) - \frac{1}{2}\cos(2x)\cos(2y). \end{aligned}$$

Original data:



Interpolant:



So far, this process is fully reversible and requires the same amount of storage (an $n \times n$ matrix). The reason the DCT is used is because it tends to concentrate the information in an image into the top-left of the matrix Y. $\longrightarrow$ The human visual system is more sensitive to these low spatial frequencies.

Basic JPEG

- In a grayscale image each pixel is represented by an integer in $[0, 255]$. (2^8, so 8-bits)
  e.g. 64×64 region requires $8(64^2) = 2^{15} = 32$ k bits of data.
- First subtract 128 from each pixel to centre around 0. $\rightarrow$ makes better use of the DCT
- Apply DCT to each 8×8 block of pixels:

$$X \to Y = C_8^{-1} X C_8 .$$

  (still fully invertible).
- To compress,

  (1) Round Y to nearest integer.

(2). Apply a "low-pass filter", e.g. set $y_{kl} = 0$ for $k + l \geq 8$ → halves storage

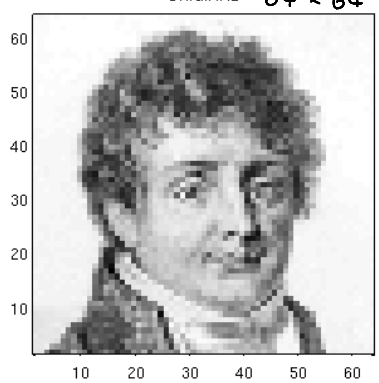
This reduces storage requirements while maintaining qualitative visual aspects of the block. ↘ next time

Remarks:-

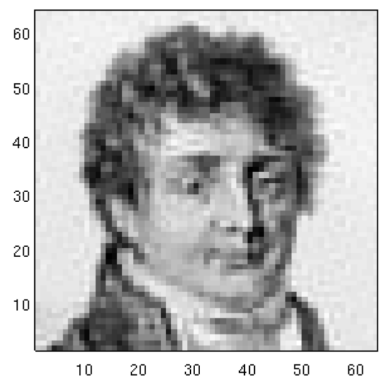
- A more sophisticated approach is to use "Hilbert qualtization" rather than low-pass filtering.
- Too much compression leads to block artefacts.
- Although sound compression is only 1-d, it is harder because the ear is more sensitive to block artefacts.
  ↪ MPEG uses a modified DCT where blocks can overlap.

Example:-

ORIGINAL 64 × 64

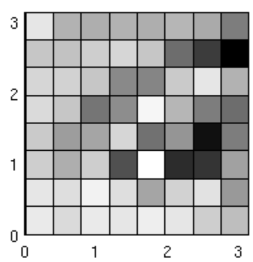


COMPRESSED → after basic JPEG

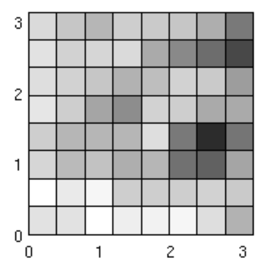


Left eye:

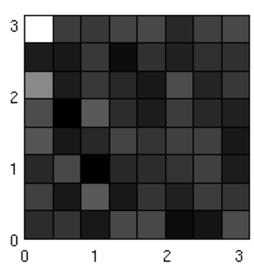
ORIGINAL

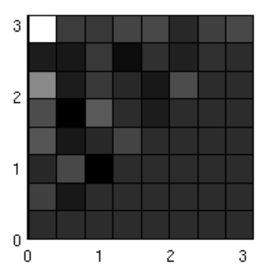


LOW-PASS FILTERED



Y





Y after filter